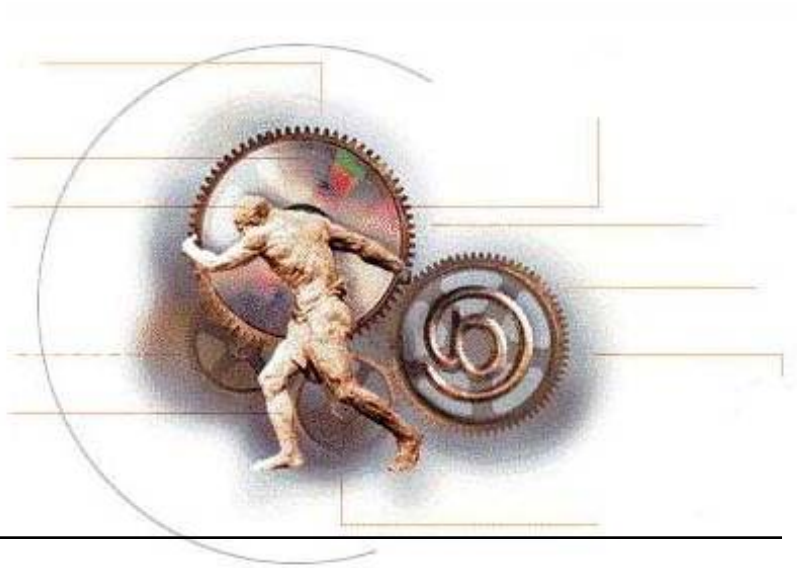


Course 0

課程介紹
Introduction

Algorithms

演算法

國立聯合大學 資訊管理學系 陳士杰老師

Outline

- 了解本課程授課重點、目標及課程設計
- 了解本課程評分標準
- 課前重點
- 資料結構 v.s. 演算法
- 資料結構概念基礎複習

教材

Text Book

- 蔡宗翰, 演算法: 使用C++ 虛擬碼, 碁峰圖書, 2004.
- 講義下載處: http://web.nuu.edu.tw/~sjchen

Reference Book

- R. E. Neapolitan and K. Naimipour, Foundations of Algorithms: Using C++ Pseudocode, 3/e, Jones, 2004. (**Textbook**原文本)
- 李家同教授, Computational Biology, 全文電子書第二章.
- 洪逸, 洪捷, 演算法-名校攻略密笈, 鼎茂圖書.
- 洪逸, 資料結構(含精選試題), 鼎茂圖書.

課程重點

- 前置觀念
  - 演算法：效率，分析與量級 (Algorithms: Efficiency, Analysis, and Order)
  - 遞迴 (Recursion)
  - 排序 (Sort)
  - 搜尋 (Search)
- 解題策略
  - Divide-and-Conquer (切割與征服)
  - Dynamic Programming (動態規劃)
  - The Greedy Approach (貪婪法則)
  - Backtracking (回溯)
  - Branch-and-Bound (分枝與限制)
- NP問題
  - An Introduction to the Theory of NP

評分標準

- 期中考：35%
- 期末考：35%
- 平時考：20%
- 平時成績：10%
- 加分程式題：10%~20%

資料結構 v.s. 演算法

資料結構

- Array
- Linked List

基礎

- Stack
- Queue
- Tree
- Graph

高階

- 複雜度分析
- 搜尋 (Search)
- 排序 (Sort)
- 高等樹 (Advanced Tree)

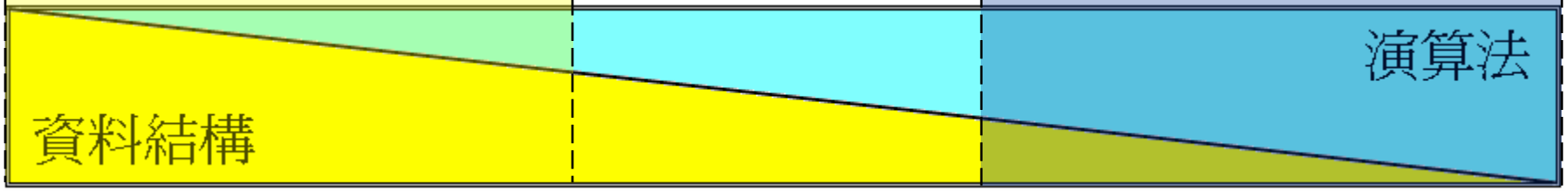
演算法

- Divide-and-Conquer
- Dynamic Programming
- The Greedy Approach
- Backtracking
- Branch-and-Bound

ADT

Theory of NP:

- P Problem
- NP Problem
- NP Complete Problem
- NP Hard Problem



資料結構概念基礎複習

Array

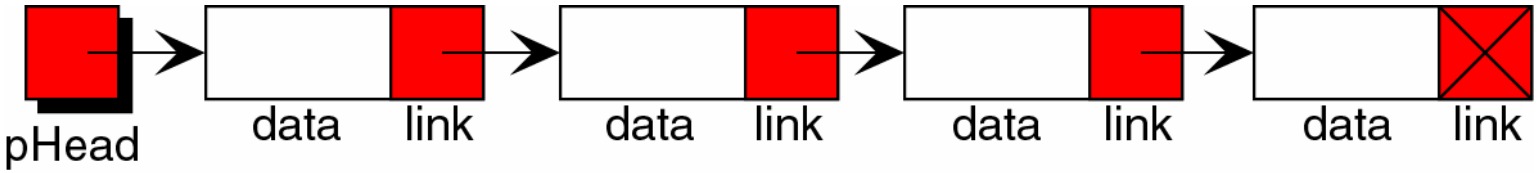
- Def:
  - 爲表示有序的線性串列之一種方式
  - 其佔用**連續性**的記憶體空間
  - 各元素型態**皆需相同** (一致)
  - 支援Sequential及Random Access
  - 插入、刪除元素較爲麻煩
    - ∵需**挪移其它元素**
    - ∴不易動態增刪空間大小

Array種類

- 一維陣列
- 二維陣列
- 三維陣列
- N維陣列

Linked List

- Def: 由一組**節點 (Node)**所組成的**有序串列**，各Node除了Data欄之外，另外有 ≥ 1 個Link欄 (或稱 Pointer)，用以指向其它Node之位址。
- 有一個指標變數 (**Pointer variable**) 用來指出鏈結串列中的第一個節點之所在。指標變數的名字可用來做爲其所指向之鏈結串列的名字。

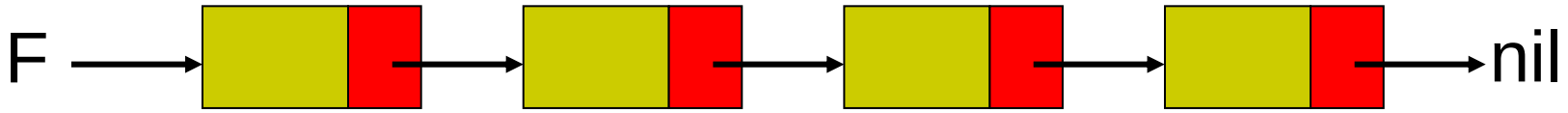


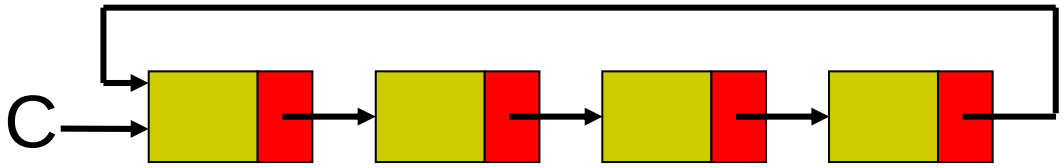
(a) A linked list with a head pointer: pHead

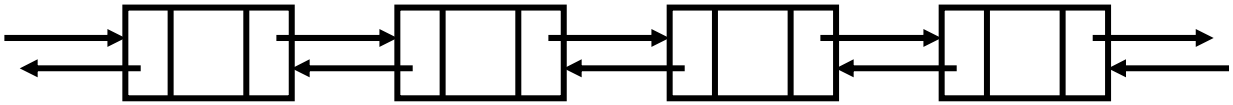
(b) An empty linked list

- Linked list的特質:
  - 各**Node**不一定要佔用連續的**Memory**空間
  - 各**Node**之型態 **(Data Type)** 不一定要相同
  - 僅支援**Sequential Access**
  - **Insert/Delete Node** 容易

Linked List 種類







Stack

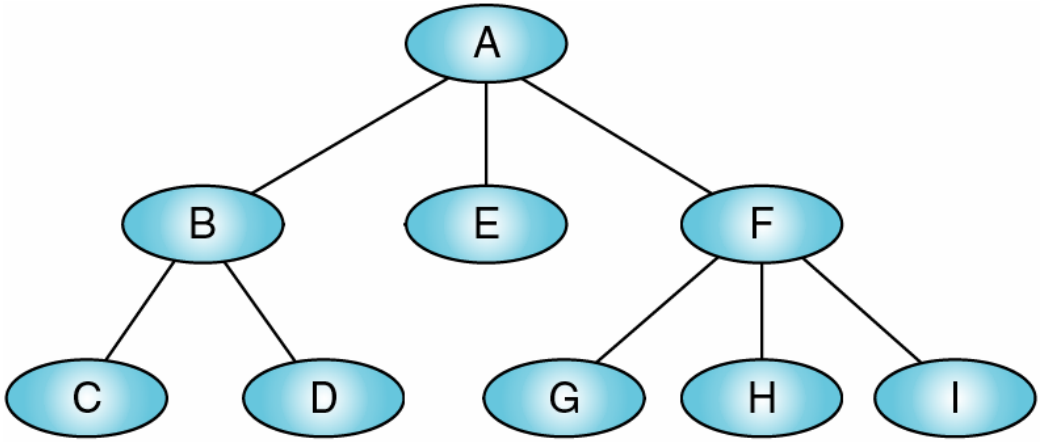
- Def: 具有**LIFO (last in-first out)**或**FILO (first in-last out)** 性質的有序串列。
  - 插入元素的動作稱爲**Push**, 刪除元素的動作稱爲**Pop**.
  - Push/Pop的動作皆發生在同一端，此端稱爲**Top**.

Queue (佇列)

- Def: 具有**FIFO (first in-first out)** 性質的有序串列。其插入元素的動作稱爲發生在**Rear (尾)端**, 刪除元素的動作發生在**Front (前)端**.

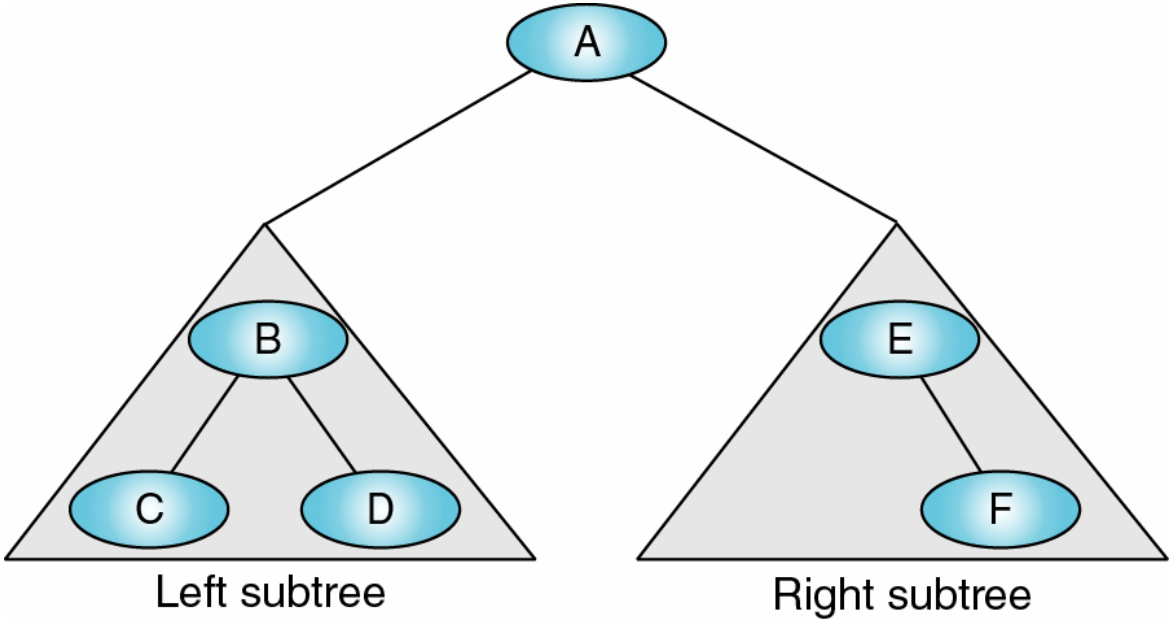
Tree

- **Def: Tree是由1個以上的Nodes所組成的有限集合，滿足:**
  - 至少有一個Node，稱爲Root
  - 其餘的Nodes分成$T_1, T_2, ..., T_n$個互斥集合，稱爲Subtree



Binary Tree (二元樹)

- Def:
  - Binary Tree爲具有 ≥ 0 個nodes所構成的有限集合。
    - Binary Tree可以爲空的樹。
    - 若不爲空的樹，則具有Root及左, 右子樹，且左, 右子樹亦是Binary Tree。



二元樹之三個基本定理

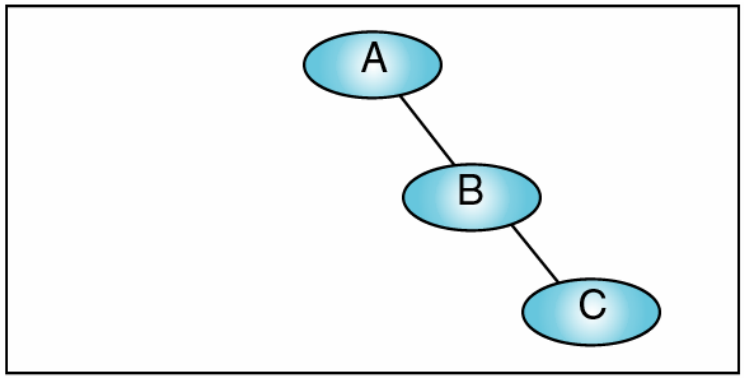
- **[定理一]:** 二元樹中，第i個level的node個數最多有 2^{i-1} 個。
- **[定理二]:** 高 (深) 度爲k的二元樹，其node個數最多有 2^k-1 個。
- **[定理三]:** 非空二元樹若leaf個數爲 n_0 個，degree爲2的node個數爲n_2個，則 $n_0 = n_2+1$。

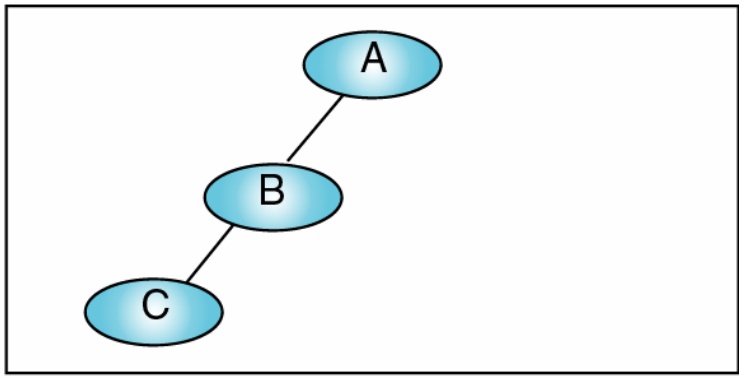
二元樹的種類

- Skewed Binary Tree
- Full Binary Tree
- Complete Binary Tree

Skewed Binary Tree

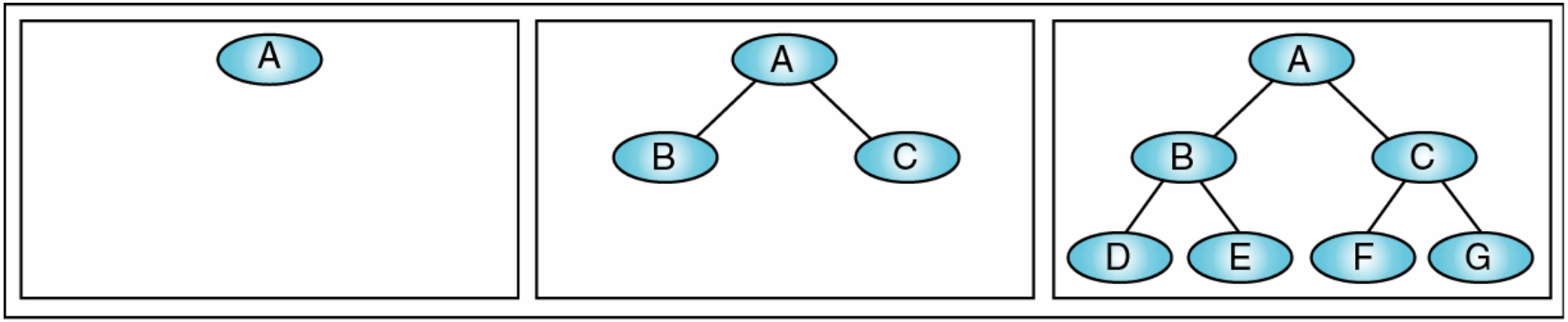
- Def: 可分爲
  - Left-Skewed Binary Tree: 每個non-leaf node皆只有左子集
  - Right-Skewed Binary Tree: 每個non-leaf node皆只有右子集





Full Binary Tree

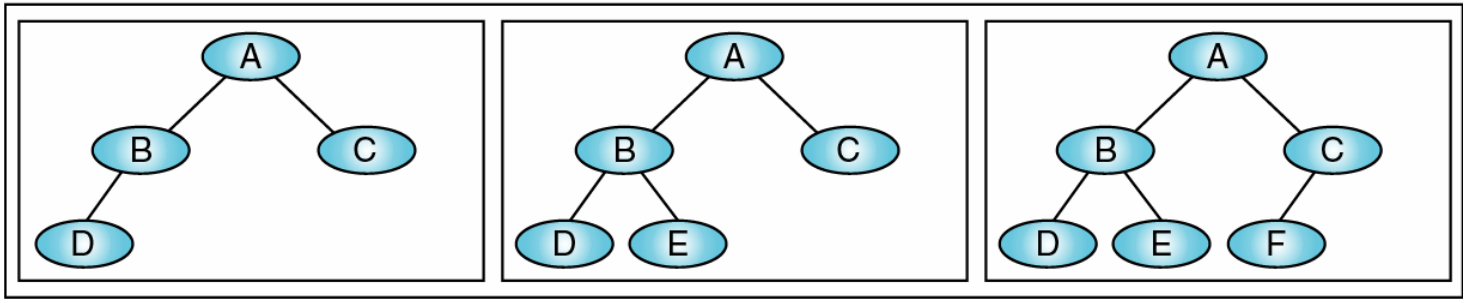
- **Def:**
  - 具有**最多Node個數**的二元樹稱之
    - 即: 高度爲d，其node個數必爲2^d-1
    - 具有n個nodes的Full B.T. ，其高度必爲: $\lceil \log_2(n+1) \rceil$



  - 課本將之命名爲: **complete binary tree (7.6.1節)**

Complete Binary Tree

- **Def:**
  - 若二元樹高度爲d，node個數爲n，則
    - $2^{d-1}-1 < n < 2^d-1$
    - n個node之編號與高度d的full binary tree之前的n個node編號一一對應，不能跳號。

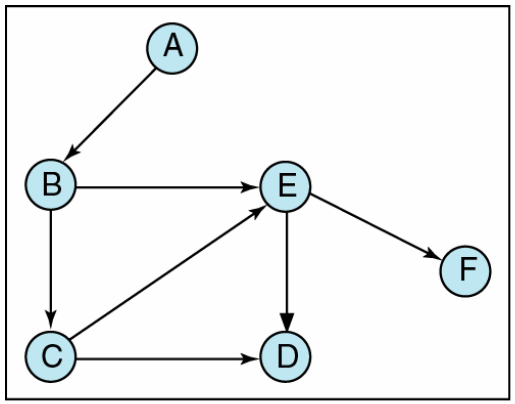


  - 課本將之命名爲: essentially complete binary tree (7.6.1節)

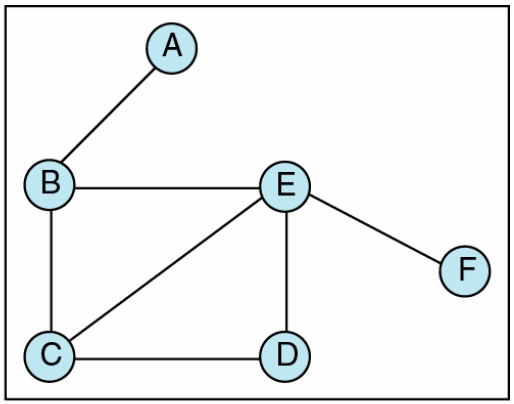
Graph

- Def:
  - A Graph is a collection of nodes, called **vertices**, and a collection of line segments, called **edges**, that connecting pairs of vertices.
  - In other words, a graph consists of two sets
    - a set of vertices
    - a set of lines.

- Graph may be either **directed** or **undirected**:
  - Directed graph (或稱 Digraph)
    - Each line has a direction to its successor.
    - G = (V, E), 其中V爲頂點集合，E爲邊集合。<vi, vj> ≠ <vj, vi>。
  - Undirected graph
    - Each line has no direction.
    - G = (V, E), 其中V爲頂點集合，E爲邊集合。<vi, vj> = <vj, vi>。



(a) Directed graph



(b) Undirected graph

Data Type and Abstract Data Type

- Data Type
  - Def: A data type consists of two parts
    - a set of **data**
    - the **operations** that can be performed on the data.
- For example:
  - Integer
    - Data: $-\infty$, ..., -2, -1, 0, 1, ..., ∞ (沒有小數的數值集合)
    - Operations: +, -, $\times$, $\div$, %, $\leq$, $\geq$, ==, !=, ++, --, ...
  - Floating point
    - Data: $-\infty$, ..., -1.9, ..., 0.0, ..., ∞
    - Operations: +, -, $\times$, $\div$, %, $\leq$, $\geq$, ==, !=, ...
  - Character
    - Data: 'A', 'B', ..., 'a', 'b', ...
    - Operations: <, >, ...

◆ ADT (Abstract Data Type; 抽象資料型態)

- **Def**: ADT是一種Data Type，且要滿足："資料的規格與操作的規格" 獨立於 "資料的實際表示方式與操作的實際製作方式."